Carnegie Mellon
Software Engineering Institute

Making Architecture Design Decisions: An Economic Approach
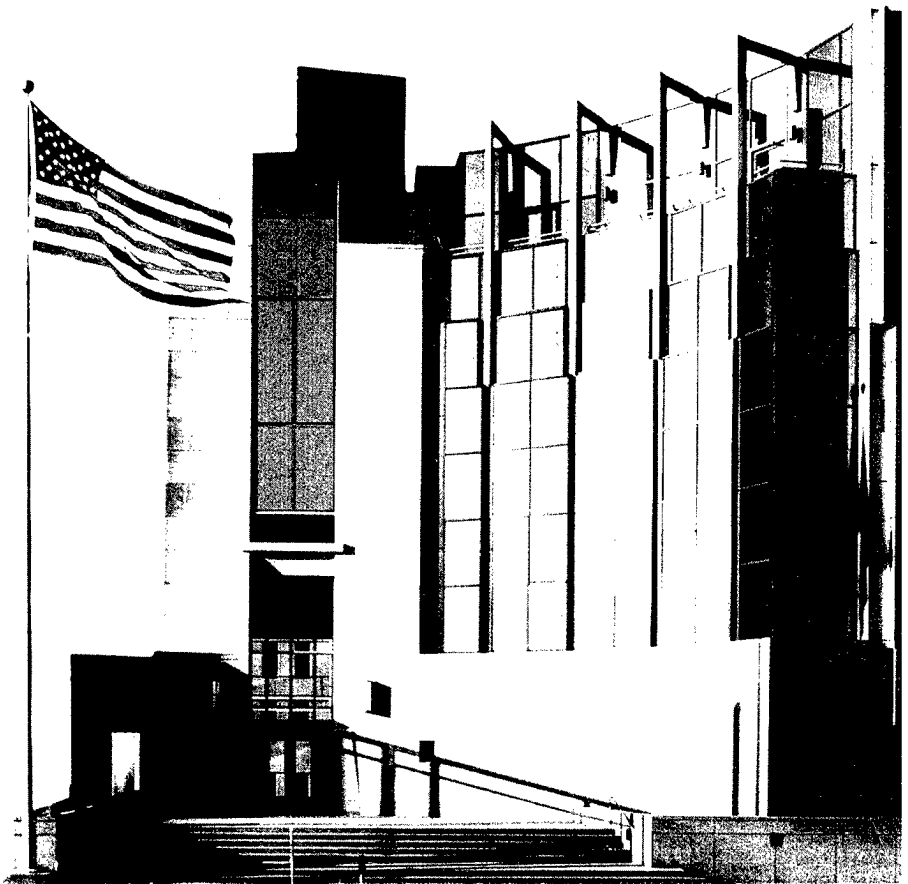
Rick Kazman
Jai Asundi
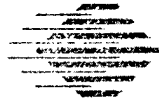Mark Klein

September 2002

20021231 103

TECHNICAL REPORT
CMU/SEI-2002-TR-035
ESC-TR-2002-035

Carnegie Mellon
Software Engineering Institute

Pittsburgh, PA 15213-3890

Making Architecture Design Decisions: An Economic Approach

CMU/SEI-2002-TR-035
ESC-TR-2002-035

Rick Kazman

Jai Asundi

Mark Klein

September 2002

Product Line Systems Program

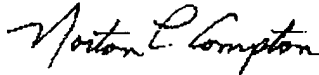*AQM03-03-0627*

This report was prepared for the

SEI Joint Program Office
HQ ESC/AXS
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER

Norton L. Compton, Lt Col, USAF
SEI Joint Program Office

Table of Contents

List of Figures

List of Tables

Abstract

The resources available to build any system are finite. The decisions involved in building any nontrivial system are complex and typically involve many stakeholders, many requirements, and many technical decisions. The stakeholders have an interest in ensuring that good design decisions are made—decisions that meet their technical objectives and their tolerance for risk. These decisions should, as much as possible, maximize the benefit that the system provides and minimize its cost. The Cost Benefit Analysis Method (CBAM) was created to provide some structure to this decision-making process. The CBAM analyzes architectural decisions from the perspectives of cost, benefit, schedule, and risk. While the CBAM does not make decisions for the stakeholders, it does serve as a tool to inform managers and to structure the inquiry so that rational decisions can be made. This report describes the steps of the CBAM and its application to a real-world system.

1 Introduction

The creation and maintenance of a complex software-intensive system involves making a series of architecture design decisions. The Architecture Tradeoff Analysis Method[SM] (ATAM[SM])[1] provides software architects with a framework for understanding the technical tradeoffs they face as they make design or maintenance decisions [Kazman 99, Clements 01]. But the biggest tradeoffs in large, complex systems usually have to do with economics, and the ATAM does not provide any guidance for understanding economic tradeoffs. Organizations need to know how to invest their resources to maximize their gains and minimize their risks. When economic issues have been addressed by others in the past, the focus has usually been on *costs*, and only on the cost of building the system, not the long-term costs of maintenance and upgrade. Yet the *benefits* that an architecture decision may bring to an organization are as important—or perhaps even more important—than the costs.

Because the resources for building and maintaining a system are finite, there must be a rational process for choosing among architectural options during the initial design phase and during subsequent periods of upgrade. These options will have different costs, consume different amounts of resources, implement different features (each bringing some benefit to the organization), and have some inherent risk or uncertainty. To explore the effects of these options, economic software models are needed that take into account costs, benefits, risks, and schedule implications.

Earlier Software Engineering Institute (SEI) technical reports describe the first version of the Cost Benefit Analysis Method (CBAM), a quantitative economic approach to making design decisions [Kazman 01, Asundi 01]. This approach asserts that architectural strategies (ASs) affect the quality attributes (QAs) of the system and in turn provide the stakeholders of the system with some benefit. The benefit added to the system through the implementation of an AS is referred to as *utility*. Each AS provides a specific level of utility to the stakeholders, and each AS has cost and schedule implications. This information can aid the stakeholders in choosing ASs based on their return on investment (ROI), which is the ratio of benefit to cost.

While this approach seems reasonable in theory, there are a number of problems with trying to implement it in practice. One problem is that QAs (such as performance, modifiability, and usability) are abstract entities to the stakeholders, making it difficult for stakeholders to inter-

---

1. Architecture Tradeoff Analysis Method and ATAM are service marks of Carnegie Mellon University.

pret QAs consistently. Stakeholders also have difficulty quantifying the expected utility level of an AS, and highly variable judgments can result if their interpretations cannot be calibrated with the current system's business goals.

We noted these problems as we prototyped the CBAM in a real-world setting, and in response we created a second version of the CBAM, which is described in this report.

This report is structured as follows: Section 2 describes the improvements to the CBAM and the steps of the new version. Sections 3 and 4 describe a pilot case study conducted with NASA, and Section 5 concludes with ideas for future work. Appendix A describes the theoretical basis of the CBAM.

1.1 Decision-Making Context

The software architect or decision maker needs to maximize the difference between the benefit derived from the system and the cost of implementing the design. The CBAM begins where the ATAM concludes and depends on the artifacts produced by the ATAM. Figure 1 depicts the context for the CBAM. Because these ASs have technical and economic implications, the business goals of a software system should influence the ASs used by software architects or designers. The *technical* implications are the characteristics of the software system (namely, QAs), while the direct *economic* implication is the cost of implementing the system. However, the QAs also have economic implications because of the benefits that can be derived from the system.
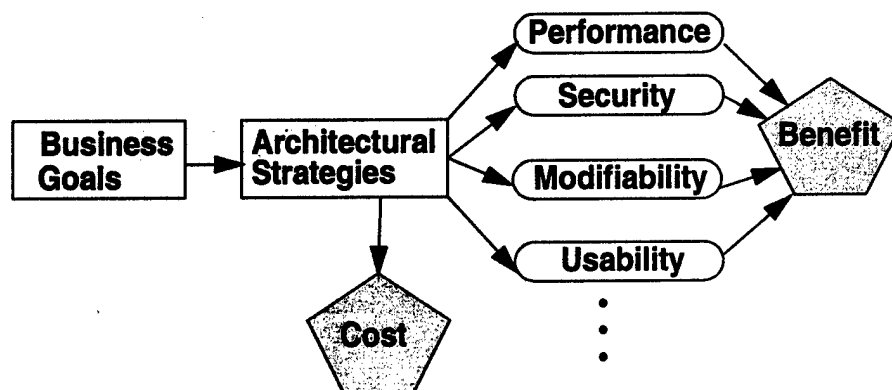
Figure 1: Context for the CBAM

When the ATAM is applied to a software system, the result is a set of documented artifacts, including

- a description of the business goals that are crucial to the system's success

- a set of architectural views that document the existing or proposed architecture

- a utility tree that represents a decomposition of the stakeholders' quality goals for the architecture, starting with high-level statements of QAs and ending with specific scenarios

- a set of architectural risks that have been identified

- a set of sensitivity points (architecture decisions that affect some QA measure of concern)

- a set of tradeoff points (architecture decisions that affect more than one QA measure, some positively and some negatively)

The ATAM identifies the set of key architectural decisions that are relevant to the QA scenarios elicited from the stakeholders. These decisions result in specific QA responses, such as levels of availability, performance, security, usability, and modifiability. But each of those architectural decisions also has associated costs. For example, using redundant hardware to achieve a desired level of availability has a cost, and checkpointing to a disk file has a different cost. Furthermore, both of these architectural decisions will result in (presumably different) measurable levels of availability with some value to the organization developing the system. Perhaps the organization believes that its stakeholders will pay more for a highly reliable system (for example, software that controls anti-lock brakes in an automobile or medical monitoring software) or that it will get sued if the system is not highly available (for example, a telephone switch).

The ATAM uncovers the architectural decisions made in the system and links them to business goals and QA response measures. The CBAM builds on this foundation by eliciting the costs and benefits associated with these decisions. Given this information, the stakeholders can decide whether to use redundant hardware, checkpointing, or some other strategy to achieve the system's desired availability. Or the stakeholders can choose to invest their finite resources in some other QA—perhaps believing that higher performance will have a better benefit to cost ratio. A system always has a limited budget for creation or upgrade, so each architectural choice is competing with all other choices for inclusion.

The CBAM does not make decisions for the stakeholders, just as a financial advisor doesn't tell clients how to invest their money. The CBAM simply aids in the elicitation and documentation of the costs, benefits, and uncertainty of a "portfolio" of architectural investments and gives the stakeholders a framework for applying a rational decision-making process that suits their needs and risk averseness.

To summarize, the idea behind the CBAM is that ASs affect the QAs of the system and in turn provide the stakeholders of the system with some level of utility. However, each AS also has costs associated with it and takes time to implement. Given this information, the CBAM can aid the stakeholders in choosing ASs based on the ROI they provide.

2　The CBAM Steps

This section first describes the steps of the CBAM and then explores how to quantify uncertainty to aid in making decisions.

2.1　Iteration I: Establish an Initial Ranking

In the first iteration of the CBAM, each step is executed to establish an initial ranking that will then be refined in the second iteration. These steps serve to reduce the size of the decision space, refine the scenarios, collect sufficient information for decision making, and establish an initial ranking of ASs.

The steps of the CBAM are as follows:

Step 1: **Collate scenarios**. Collate the scenarios elicited during the ATAM exercise, and allow stakeholders to contribute new ones. Prioritize these scenarios based on satisfying the business goals of the system and choose the top one-third for further study.

Step 2: **Refine scenarios**. Refine the scenarios by focusing on their stimulus/response measures. Elicit the worst, current, desired, and best-case QA response level for each scenario.

Step 3: **Prioritize scenarios**. Allocate 100 votes to each stakeholder and have them distribute the votes among the scenarios by considering the *desired* response value for each scenario. Total the votes and choose the top 50% of the scenarios for further analysis. Assign a weight of 1.0 to the highest rated scenario, and assign the other scenarios a weight relative to that scenario. This becomes the scenario weight that appears in the calculation of the overall benefit of an AS. Make a list of the QAs that concern the stakeholders.

Step 4: **Assign utility**. Determine the utility for each QA response level (worst-case, current, desired, or best-case) for all scenarios. The QAs of concern are the ones in the list generated in step 3.

Step 5: **Develop ASs for scenarios and determine their expected QA response levels.** Develop (or capture the already-developed) ASs that address the top 50% of the scenarios chosen in step 3, and determine the expected QA response levels that will result from implementing these ASs. Given that an AS may affect multiple scenarios, this calculation must be performed for each affected scenario.

Step 6: **Determine the utility of the expected QA response level by interpolation.** Using the elicited utility values (that form a utility curve) determine the utility of the "expected" QA response level. We perform this calculation for each affected scenario.

Step 7: **Calculate the total benefit obtained from an AS.** Subtract the utility value of the "current" level from the "expected" level and normalize it using the votes elicited in step 3. Sum the benefit of a particular AS across all scenarios and across all relevant QAs.

Step 8: **Choose ASs based on ROI, subject to cost and schedule constraints.** Determine the cost and schedule implications of each AS. Calculate the ROI value for each remaining AS as a ratio of benefit to cost. Rank order the ASs according to their ROI values, and choose the top ones until the budget or schedule is exhausted.

Step 9: **Confirm the results with intuition.** Of the chosen ASs, consider whether these seem to align with the organization's business goals. If not, consider issues that may have been overlooked during the analysis. If significant issues exist, perform another iteration of these steps.

2.2 Iteration II: Incorporating Uncertainty

A more sophisticated and realistic version of the CBAM can be created by expanding on the steps enumerated above. Secondary information can be added about risk estimation and uncertainty and the allocation of development resources. Each category of secondary information may potentially affect the investment decisions under consideration. Therefore, the ways they augment the steps of the method must be considered carefully for correctness and for practicality.

2.2.1 Augmenting the Steps

The CBAM relies on stakeholder judgments for its valuations of software benefits and costs. But these judgments will naturally be uncertain, due to differences in beliefs and experiences. One way to think rigorously about the uncertainty of the results collected in Iteration I is to collect and consider the risks inherent in the estimates that have been made. To do this, some kind of risk assessment exercise must be performed. The risks typically fall into these four categories:

1. risks that affect the cost estimate of a strategy under consideration

2. risks that affect a stimulus-response characteristic or a utility estimate of a strategy in the context of a scenario

3. risks that affect stimulus-response characteristics of other scenarios or QAs not previously considered. These risks pertain to the side effects (rather than the intended effects) of an AS.

4. risks that are related to project management and schedule

Given this risk information, it can no longer be assumed that the costs of ASs, the resulting QA response levels, and the associated utility levels are all known and elicited from the stakeholders with certainty; therefore, the uncertainty along those three dimensions can be dealt with explicitly. The final result of this iteration is to associate with each cost, response, and utility a *range* (distribution) of values, rather than a single point value for each.

The stakeholders need to assess the probability and impact of each risk and use this to create a delta expected cost, delta expected response, and utility value for each risk affecting each AS. When this technique has been employed, the result is a range of values rather than a single point. Probability calculations are then used to determine the likelihood of the ROI ranking (determined in step 8) changing. This calculation is discussed in the appendix.

2.3 Process Steps and Triage

A process flow diagram for the CBAM can now be created; an example is depicted in Figure 2. The first four steps in this figure are annotated with the relative numbers of scenarios that they consider. The number of scenarios considered steadily decreases. This is the CBAM's way of concentrating the stakeholders' limited time and attention on the scenarios that are believed to be of the greatest potential in terms of ROI.

Any method for improving the quality of a system, or for making strategic decisions about the system, has a cost: the cost of running the method. It also has some benefits: better, more informed decisions, a formal process that the stakeholders follow, and greater stakeholder buy-in. Presumably the stakeholders would like to maximize the ROI of running the CBAM just as much as they would like to maximize the ROI of the system itself. For this reason, it is critical that the time spent on the CBAM is kept to a reasonable level and is focused on the most important scenarios and ASs.
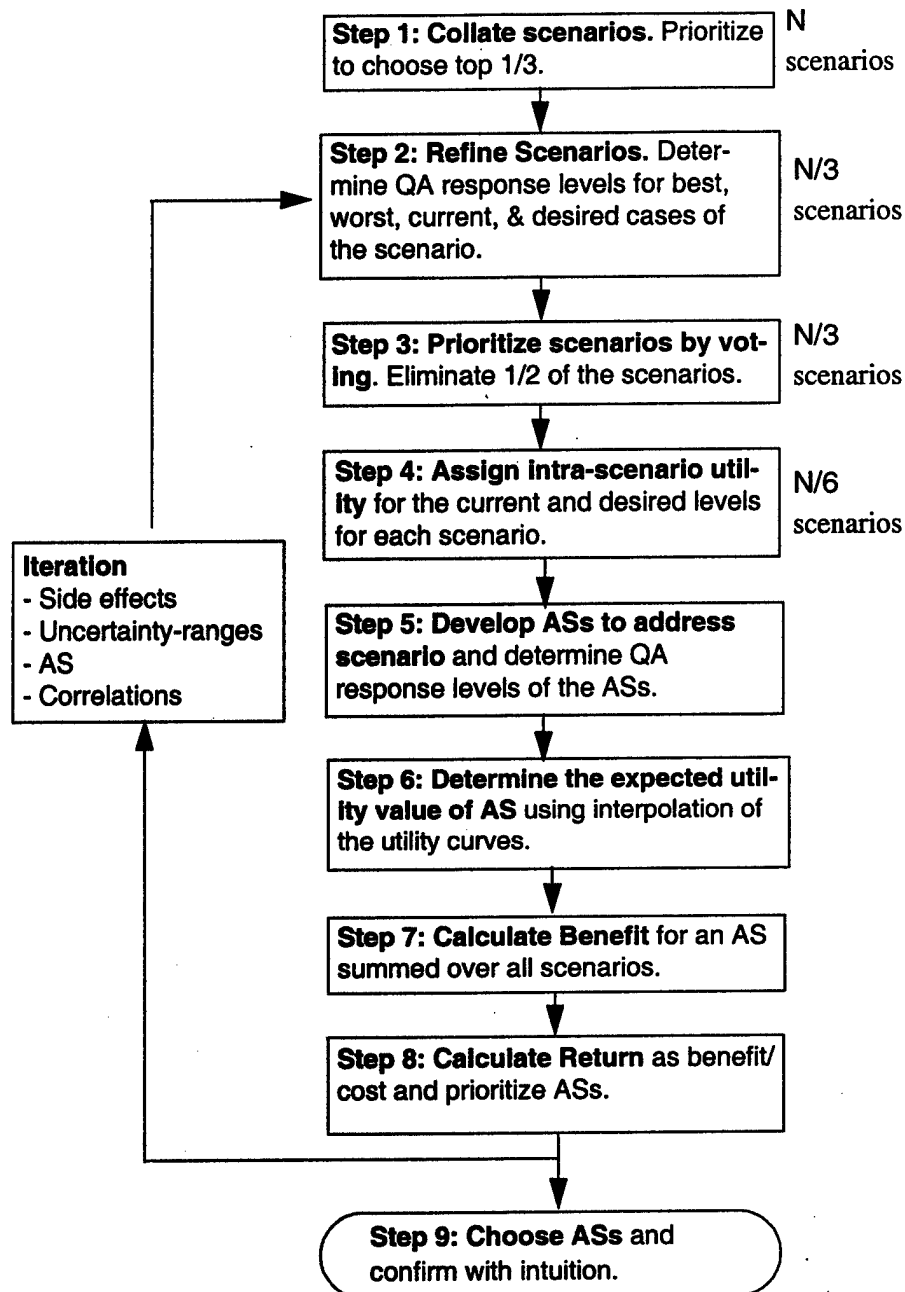
Figure 2: Process Flow Diagram for the CBAM

3 Case Study: The NASA ECS Project— Iteration I

The following case study shows how the CBAM has been applied to a real-world system.

The Earth Observing System is a constellation of NASA satellites that gathers data about the Earth for the U.S. Global Change Research Program and other scientific communities worldwide. The Earth Observing System Data Information System (EOSDIS) Core System (ECS) collects data from various satellite downlink stations for further processing. The mission of the ECS is to process the data into higher-form information and make it available in searchable form to scientists around the world. The goals are to provide a common way to store and process data and to provide a public mechanism for introducing the new data formats and processing algorithms needed, thus making the information widely available to the scientific community at large. The ECS processes an input stream of hundreds of gigabytes of raw environment-related data per day. Processing involves the computation of 250 standard "products" and results in thousands of gigabytes of information that get archived at 8 data centers in the U.S. The ECS has important performance and availability requirements. The long-term nature of the project also makes modifiability an important requirement.

The project manager at ECS had a limited annual budget to maintain and enhance his current system. During a prior analysis (in this case an ATAM exercise), many desirable changes to the system were elicited from the stakeholders, resulting in a large set of ASs.[2] The manager had to choose a much smaller subset of ASs for implementation, since only a subset of the proposed changes could actually be funded. The manager used the CBAM to help make a rational decision based on ROI.

3.1 Step 1: Collate Scenarios

Scenarios from the ATAM were collated with a set of new scenarios elicited from the assembled stakeholders. Because the ECS stakeholders had previously been through an ATAM exercise for ECS, this step was relatively straightforward for them.

---

2. The ATAM does not ordinarily produce a set of ASs, but in this case it did.

The raw scenarios are shown in Table 1. Note that the scenarios are not yet well-formed and that some of them do not have defined responses. These issues are resolved in step 2 when the number of scenarios is reduced.

Table 1: *Collected Scenarios*

Scenario	Scenario Description
1	Reduce data distribution failures that result in hung distribution requests requiring manual intervention.
2	Reduce data distribution failures that result in lost distribution requests.
3	Reduce the number of orders that fail on the order submission process.
4	Reduce order failures that result in hung orders requiring manual intervention.
5	Reduce order failures that result in lost orders.
6	There is no good method of tracking ECS. Guest failed/cancelled orders without much manual intervention (e.g., spreadsheets).
7	Users need more information on why their orders for data failed.
8	Due to limitations, there is a need to artificially limit the size and number of orders.
9	Small orders result in too many notifications to users.
10	The system should process a 50 gigabyte user request in 1 day, and a 1 terabyte user request in 1 week.

3.2   Step 2: Refine Scenarios

These scenarios were then refined, paying particular attention to specifying their stimulus/ response measures precisely. The worst case, the current case, the desired case, and the best case response goal for each scenario were elicited and recorded, as shown in Table 2.

Table 2: *Response Goals for Refined Scenarios*

	Response Goals			
Scenario	Worst	Current	Desired	Best
1	10% hung	5% hung	1% hung	0% hung
2	> 5% lost	< 1% lost	0% lost	0% lost
3	10% fail	5% fail	1% fail	0% fail
4	10% hung	5% hung	1% hung	0% hung
5	10% lost	< 1% lost	0% lost	0% lost
6	50% need help	25% need help	0% need help	0% need help
7	10% get info	50% get info	100% get info	100% get info
8	50% limited	30% limited	0% limited	0% limited
9	1 per granule	1 per granule	1 per 100 granules	1 per 1000 granules
10	< 50% meet goal	60% meet goal	80% meet goal	> 90% meet goal

3.3 Step 3: Prioritize Scenarios

The team voted on the refined representation of the scenarios. This close-knit group chose to deviate slightly from the method by not voting individually, but instead discussing each scenario and determining its weight by consensus. The votes allocated to the entire set of scenarios were constrained to a total of 100, as shown in Table 3. Although the stakeholders were not required to make the votes multiples of five, they felt that this was reasonable, and that more precision in the votes was not needed and could not be justified.

Table 3:   Refined Scenarios with Votes

Scenario	Votes	Response Goals			
		Worst	**Current**	**Desired**	**Best**
1	10	10% hung	5% hung	1% hung	0% hung
2	15	> 5% lost	< 1% lost	0% lost	0% lost
3	15	10% fail	5% fail	1% fail	0% fail
4	10	10% hung	5% hung	1% hung	0% hung
5	15	10% lost	< 1% lost	0% lost	0% lost
6	10	50% need help	25% need help	0% need help	0% need help
7	5	10% get info	50% get info	100% get info	100% get info
8	5	50% limited	30% limited	0% limited	0% limited
9	10	1 per granule	1 per granule	1 per 100 granules	1 per 1000 granules
10	5	< 50% meet goal	60% meet goal	80% meet goal	> 90% meet goal

3.4 Step 4: Assign Utility

In this step the utility for each scenario was determined by the stakeholders, again by consensus. A utility score of 0 represents no utility; a score of 100 represents the most utility possible. The results of this process are given in Table 4.

Table 4:   Scenarios with Votes and Utility Scores

Scenario	Votes	Utility Scores			
		Worst	**Current**	**Desired**	**Best**
1	10	10	80	95	100
2	15	0	70	100	100
3	15	25	70	100	100
4	10	10	80	95	100
5	15	0	70	100	100

Table 4:   Scenarios with Votes and Utility Scores (Continued)

Scenario	Votes	Utility Scores			
		Worst	Current	Desired	Best
6	10	0	80	100	100
7	5	10	70	100	100
8	5	0	20	100	100
9	10	50	50	80	90
10	5	0	70	90	100

3.5   Step 5: Develop ASs for Scenarios and Determine Their Expected Response Levels

Based on the requirements implied by the above scenarios, a set of 10 ASs was developed by the ECS architects. Recall that an AS may affect more than one scenario. To account for these complex relationships, the expected QA response level that each AS was predicted to achieve had to be determined with respect to each relevant scenario.

The set of ASs is shown in Table 5, along with the scenarios they address. For each AS/scenario pair, the response levels that the AS is expected to achieve with respect to that scenario is shown (along with the current response, for comparison purposes).

Table 5:   Architectural Strategies and Scenarios Addressed

AS	AS Name	AS Brief Description	Scenarios Affected	Current Response	Expected Response
1	Order persistence on submission	Store order as soon as it arrives in system.	3	5% fail	2% Fail
			5	<1% lost	0% lost
			6	25% need help	0% need help
2	Order chunking	Allow operators to partition large orders into multiple small orders.	8	30% limited	15% limited
3	Order bundling	Combine multiple small orders into one large order.	9	1 per granule	1 per 100
			10	60% meet goal	55% meet goal
4	Order segmentation	Allow an operator to skip items that cannot be retrieved due to data quality or availability issues.	4	5% hung	2% hung
5	Order reassignment	Allow an operator to reassign the media type for items in an order.	1	5% hung	2% hung

Table 5: Architectural Strategies and Scenarios Addressed (Continued)

AS	AS Name	AS Brief Description	Scenarios Affected	Current Response	Expected Response
6	Order retry	Allow an operator to retry an order or items in an order that may have failed due to temporary system or data problems.	4	5% hung	3% hung
7	Forced order completion	Allow an operator to override an item's unavailability due to data quality constraints.	1	5% hung	3% hung
8	Failed order notification	Ensure that users are notified only when part of their order has truly failed and provide a detailed status of each item; user notification would occur only if the operator okays such a notification; the operator may edit the notification.	6	25% need help	20% need help
			7	50% get info	90% get info
9	Granule-level order tracking	An operator and user can determine the status for each item in their order.	6	25% need help	10% need help
			7	50% get info	95% get info
10	Links to user information	An operator can quickly locate a user's contact information. The server will access science data server (SDSRV) information to determine any data restrictions that might apply and will route orders/order segments to the appropriate distribution capabilities, including data distribution (DDIST), PDS, external subsetters, and data processing tools.	7	50% get info	60% get info

3.6 Step 6: Determine Expected Utility Levels by Interpolation

Now that the expected response level of each AS has been characterized with respect to a set of scenarios, the utility of these expected response levels can be calculated by consulting the utility scores for each scenario's current and desired responses for all of the affected attributes. Using these scores the utility of the expected QA response levels for the AS/scenario pair can be calculated through interpolation or extrapolation.

Table 6 shows the results of this calculation for the AS/scenario pairs presented in Table 5.

Table 6: Architectural Strategies and Their Expected Utility

AS	AS Name	Scenarios Affected	Current Utility	Expected Utility
1	Order persistence on submission	3	70	90
		5	70	100
		6	80	100
2	Order chunking	8	20	60
3	Order bundling	9	50	80
		10	70	65
4	Order segmentation	4	80	90
5	Order reassignment	1	80	92
6	Order retry	4	80	85
7	Forced order completion	1	80	87
8	Failed order notification	6	80	85
		7	70	90
9	Granule-level order tracking	6	80	90
		7	70	95
10	Links to user information	7	70	75

3.7 Step 7: Calculate the Total Benefit Obtained from an AS

Based on the information collected, as represented in Table 6, the total benefit of each AS can be calculated using Equation 1 from Section A.5. For each AS, this equation calculates the total benefit as the sum of the benefit that accrues to each scenario, normalized by the relative weight of the scenario. The total benefit scores for each AS are given in Table 7.

Table 7: Total Benefit of Architectural Strategies

Strategy	Scenario Affected	Scenario Weight	Raw AS Benefit	Normalized AS Benefit	Total AS Benefit
1	3	15	20	300	
1	5	15	30	450	
1	6	10	20	200	950
2	8	5	40	200	200
3	9	10	30	300	
3	10	5	-5	-25	275
4	4	10	10	100	100
5	1	10	12	120	120
6	4	10	5	50	50

Table 7:   Total Benefit of Architectural Strategies (Continued)

Strategy	Scenario Affected	Scenario Weight	Raw AS Benefit	Normalized AS Benefit	Total AS Benefit
7	1	10	7	70	70
8	6	10	5	50	
8	7	5	20	100	150
9	6	10	10	100	
9	7	5	25	125	225
10	7	5	5	25	25

3.8   Step 8: Choose ASs Based on ROI Subject to Cost Constraints

To complete the analysis, a cost estimate was done for each AS. The team provided cost estimates based on its experience with the system, and an ROI was calculated for each AS based on those estimates. Using the ROI, each of the ASs was ranked, as shown in Table 8. The ranks roughly follow the ordering that the ASs proposed: AS 1 has the highest rank, with AS 3 second. AS 9 has the lowest rank, and AS 10 has the second lowest rank. This validates the stakeholders' intuition about which ASs were most beneficial.

Table 8:   ROI of Architectural Strategies

AS	Cost	Total AS Benefit	AS ROI	AS Rank
1	1200	950	0.79	1
2	400	200	0.5	3
3	400	275	0.69	2
4	200	100	0.5	3
5	400	120	0.3	7
6	200	50	0.25	8
7	200	70	0.35	6
8	300	150	0.5	3
9	1000	225	0.22	10
10	100	25	0.25	8

4 Case Study: The NASA ECS Project— Iteration II

In Iteration II, the team discussed many important risk factors as a group. These factors were not given significant consideration in Iteration I due to time constraints and the triage focus of that iteration.

Each risk pertained to 1 or more of the 10 ASs that had been proposed previously, as shown in Table 9. For each risk factor, the following topics were discussed and assessed:

- the probability of occurrence (as an estimated probability score)
- the impact it would have if it did occur (in terms of changes to the response level and the resultant changes in utility)
- mitigation strategies

Table 9: *Risks for the Various Architectural Strategies*

ID	Risk/ AS Description	Probability	Impact	Mitigation Strategy
AS1: Order persistence on submission				
T4	Selected database management system (DBMS) may become unavailable. (T4)	0.4	High, cost increase of up to 50%	Replace DBMS; cost increase equivalent to 50% source lines of code (SLOC).
T5	Ability to store complex order objects in DBMS may not support spatial attributes as advertised. (T5)	0.5	Medium, operability and user satisfaction decrease	Do not support spatial attributes.
T6	DBMS vendor change may significantly alter new object model performance characteristics. (T6)	0.3	High, cost increase of up to 70%	Write code to use non-object elements of DBMS; cost increase equivalent to 70% SLOC.
S1	New object interface to DBMS may require far more code than anticipated to develop strategy. (S1)	0.6	High, cost increase by up to 70%	none
S2	DBMS vendor may issue patch that must be incorporated late during strategy testing. (S2)	0.7	Low, cost increase by 25%	none
AS2: Order chunking	none			
AS3: Order bundling	none			

Table 9:   Risks for the Various Architectural Strategies (Continued)

ID	Risk/ AS Description	Probability	Impact	Mitigation Strategy
AS4: Order segmentation				
T2	The implementation could result in extensive operations overhead to segment orders, thus reducing operability.	0.5	High, operability decrease	Redesign interface; 50% new SLOC.
AS5: Order reassignment				
T1	Fewer than expected hung orders may benefit from the ability to reassign the order. (T1)	0.7	Medium, 3% hung	Place more emphasis on forced order completion.
AS6: Order retry				
S3	S3: Benefit of retrying orders may change due to changing archive media over next two years.	0.3	Low, current may change to goal	none
AS7: Forced order completion				
AS8: Failed order notification				
AS9: Granule level order tracking				
T3	Team may not have adequate user interface experience to provide an acceptable user interface without extensive iterations with users. (T3)	0.8	High, cost increase of up to 50%	Add specialized UI expertise; cost increase equivalent to 20% SLOC.
AS10: Links to user information				
T7	Users may decide they want links to information not currently planned in strategy.	0.5	Low to Medium, cost increase by 20 to 50%	Write code to support additional information; cost increase of 20 to 50%, depending on number of items.
C1	Web server licensing scheme may change to require a per-user license.	0.4	High, cost could quadruple	Modify system to use alternate Web server; cost equivalent to 200% of SLOC.
C2	Advanced Web development expertise may be required.	0.5	Medium, cost increase of 50%	Add specialized UI expertise; cost increase equivalent to 20% SLOC.

Next, the risks were discussed with respect to the categorization presented in Section 2.2.1 to indicate whether they affected cost, benefit, or schedule, and whether they had any side effects. The results of this categorization are shown in Table 10. In some cases, a risk was categorized as having a primary and a secondary effect (indicated by appending an "s" to the risk ID).

Table 10:   Risks and Their Categories

ID	Type	Comment
T4	I	Cost
T5	III	Benefit - Side effect
T6	I	Cost
S1	I	Cost
S2	I	Cost
T2	III	Benefit - Side effect
T1	II	Benefit
S3	II	Benefit
T3	I	Cost
T3s	IV	Schedule
T7	I	Cost
C1	I	Cost
C2	I	Cost
C2s	IV	Schedule

Finally, the impacts of the risks and the probabilities of the events were explicitly determined by the stakeholders, and an expected risk-based delta cost (raising the maximum cost) and delta benefit (lowering the minimum benefit) were calculated. Based on these elicited values, the benefits and costs were re-estimated using a pessimistic estimate for the spread of the uniform distribution for the benefit and the cost.[3] In addition, for the cost estimate, a 10% factor was added to account for the error in the underlying cost estimation process.

Based on this exercise a range of values was computed for the cost and for the benefit. The minimum and maximum values are the end points of the confidence intervals. The expected return (E(Return)) is simply calculated as the mean of the minimum and maximum return values. The results of these new cost calculations and the benefit elicitation exercise are collated in Table 11.

Table 11:   Uncertainty Estimates Using Ranges

AS #	Cost		Benefit		Return		E(Return)	RANK
	Min	Max	Min	Max	Min	Max		
1	1200	7152.75	850	950	0.119	0.79	0.454	4
2	360	440	180	220	0.409	0.611	0.510	2
3	360	440	247.5	302.5	0.563	0.84	0.701	1

---

3.   In a pessimistic estimate, benefit/cost has the maximum spread. The largest value possible is taken to be the maximum value, and the smallest value possible is the minimum.

Table 11: Uncertainty Estimates Using Ranges (Continued)

AS #	Cost		Benefit		Return		E(Return)	RANK
4	180	220	50	100	0.227	0.556	0.391	5
5	360	440	80	120	0.182	0.333	0.258	7
6	180	220	0	50	0	0.278	0.139	10
7	180	220	63	77	0.286	0.428	0.357	6
8	270	330	135	165	0.409	0.611	0.510	2
9	900	1650	202.5	247.5	0.123	0.275	0.191	8
10	90	440	22.5	27.5	0.051	0.306	0.166	9

Based on the above probability functions, a matrix can be produced that shows the confidence in the earlier rank orderings (from Table 8) given the additional information collected regarding the uncertainties. This matrix is shown in Table 12.

Table 12: Probability That AS(row) > AS (column), Given a Uniform Distribution

	AS1	AS2	AS3	AS4	AS5	AS6	AS7	AS8	AS9	AS10	AVG
AS1	0.5	0.49	0.23	0.64	0.82	0.94	0.69	0.49	0.89	0.91	0.68
AS2	0.51	0.5	0.02	0.84	1	1	0.99	0.5	1	1	0.76
AS3	0.77	0.98	0.5	1	1	1	1	0.98	1	1	0.97
AS4	0.36	0.16	0	0.5	0.89	0.99	0.6	0.16	0.99	0.98	0.57
AS5	0.18	0	0	0.11	0.5	0.89	0.05	0	0.87	0.87	0.46
AS6	0.06	0	0	0.01	0.11	0.5	0	0	0.31	0.4	0.10
AS7	0.31	0.01	0	0.4	0.95	1	0.5	0.01	1	1	0.52
AS8	0.51	0.5	0.02	0.84	1	1	0.99	0.5	1	1	0.76
AS9	0.11	0	0	0.01	0.13	0.69	0	0	0.5	0.62	0.17
AS10	0.09	0	0	0.02	0.13	0.60	0	0	0.38	0.5	0.13

As shown in Table 12, the probability that AS1>AS3 = 0.23, so there is only a 23% chance that the rank ordering decision that AS3>AS1 is incorrect. In this manner, the confidence of the rank ordering can be judged.

Managers viewing these rankings and their associated probability values can choose to commit to one or more ASs, or they can choose to spend more effort getting better estimates of costs and benefits if the two attractive alternatives are too close to differentiate. (The second choice is particularly desirable when the probability that one is greater than the other is between 0.4 and 0.6.)

5 Conclusions and Future Work

In our experiences with the CBAM, we have discovered that solving a problem in theory is very different from solving one in practice. We have already modified the CBAM considerably as a result of the two applications of this method to the ECS project. But we also recognize that we need even more experience in applying these techniques in practice, particularly on other projects, to guide us in optimizing the method.

The new version of the CBAM is an iterative elicitation process combined with a decision analysis framework. It incorporates scenarios to represent the various QAs. The stakeholders explore the decision space by eliciting utility-QA level curves to understand how the system's utility varies with changing attributes. The consensus basis of the method allows for active discussion and clarification among the stakeholders. The traceability of the design decision permits updating and continuous improvement of the design process over time.

The CBAM has several benefits. First, it forces the stakeholders to turn their intuition about costs and benefits into explicit judgments—judgments that are based on business goals and whose underlying assumptions are discussed openly. Second, the method uses ROI as the measure of the AS's merit, rather than simply relying on either cost or benefit for the ranking. Third, and perhaps most important, the method forces the stakeholders to compare different response values using a common coin—utility—that is related to the achievement of the project's business goals.

Clearly, as software becomes even more important to the economics of many kinds of organizations, it is in their own self-interest to increase their institutional capability through the use of economic modeling in their decision-making process. Understanding the relationship between their business goals, QAs, and the utility they derive from them is an important (and fairly simple) first step. Over time, if an organization were to develop models (even heuristic ones) relating a QA response to utility or benefit, then the elicitation exercise would be much easier and less subjective.

Our experience with the CBAM tells us that giving people the appropriate tools to frame and structure their decision-making process in relation to costs and benefits and encouraging the right kind of dialogue among the stakeholders are beneficial to the development of the software system.

Appendix A   The Basis for the CBAM

This appendix describes the key ideas that form the theoretical basis for the CBAM, with a goal of explicating the theory underpinning the creation of ROI as a measure for various ASs through the use of scenarios chosen by stakeholders.

First, a collection of scenarios is considered and the different values of their projected responses are examined. Utility is then assigned to those various values of the projected responses, based on the importance of each scenario with respect to its expected response value. Next, the ASs that led to the various projected responses are considered. Each of these ASs is examined in two ways: first by the cost and then by the way in which the strategy impacts multiple QAs. That is, an AS could be used to achieve some projected response, but by achieving that response, it might also effect another QA. The utility of these "side effects" must be considered when calculating the overall utility of an AS. It is this overall utility that is combined with the project cost of an AS to calculate a final ROI measure.

A.1   Explicitly Characterized Scenarios

To deal consistently with the problem of quantifying and understanding the impact of the ASs on the QAs, the stakeholders need a concrete representation of the QAs. Scenarios elicited during the ATAM can be used for this purpose. Rather than asking the stakeholders to interpret a vague concept such as security or performance, these concepts are made more concrete through the use of specific security or performance scenarios. These scenarios specify stimulus-response characteristics and provide a tangible example that the stakeholders can use to rate the effect of each AS. The scenarios also provide a basis for deriving system utility and make it possible to rate the relative importance of the ASs.

For example, the term *performance* may mean different things to different stakeholders, but the scenario "the server component should process at least 200 transactions per minute in a steady state with an average latency of 2.5 seconds" provides little room for subjective interpretation. Every scenario, as envisioned by the ATAM, has a stimulus, a response, and an environment. In the above scenario, the stimulus is 200 transactions per minute, the environment is the steady state, and the response is stipulated at an average latency of 2.5 seconds. Every change could potentially affect the stimulus-response characteristics of a scenario in desirable

or undesirable ways. Therefore, the scenario becomes the basis for estimating the effects of an AS.

A.2   Utility-Response Curves

While the ATAM uses individual scenarios, the CBAM uses a set of scenarios (generated by varying the values of the responses). This leads to the concept of a utility-response curve. Every stimulus-response value pair in a scenario provides some utility to the stakeholders, and the utility of the different possible values for the response can be compared. For example, a very high availability in response to a failure scenario might be valued by the stakeholders only slightly more than a moderate availability level. But a low latency response to a particular performance scenario might be valued substantially more than a latency that is moderate. Each relationship between a set of utility measures and a corresponding set of response measures can be portrayed as a graph—a utility-response curve. Some samples of utility-response curves are shown in Figure 3. In each example, the point labeled "a" represents one value of the response, and the point labeled "b" represents a different value. The utility-response curve thus shows the utility as a function of the response value.
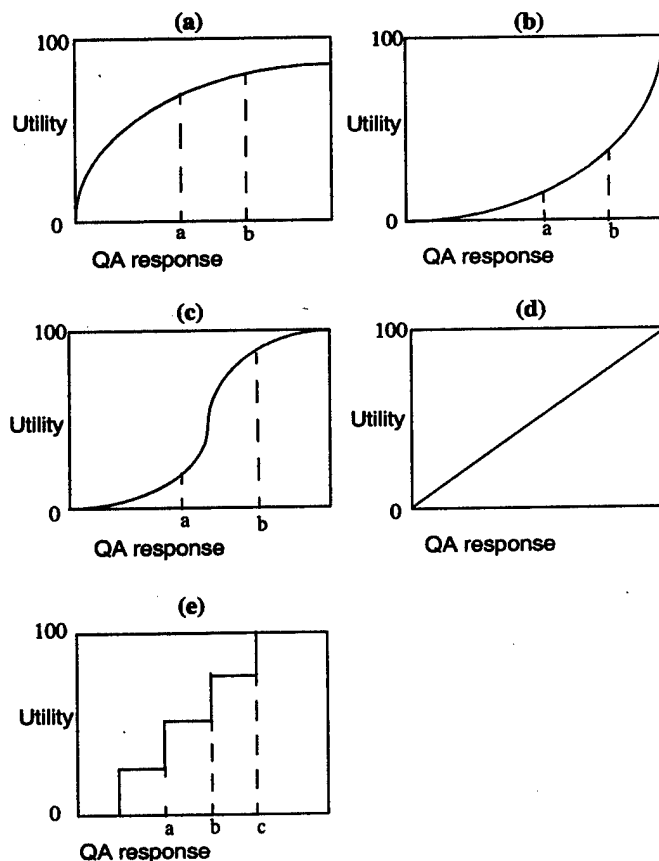

Figure 3:   Some Sample Utility-Response Curves

The utility-response curve depicts how the utility derived from a particular response varies as the response varies. As seen in Figure 3, the utility could vary non-linearly, linearly, or even as a step function. For example, graph (c) portrays a steep rise in utility over a narrow change in a QA response level, such as the performance example stated above. The availability example might be better characterized by graph (a), where a modest change in the response level results in only a very small change in utility to the user.

Eliciting the utility characteristics from the stakeholders can be a long and tedious process. To make this process practical, the stakeholders are asked only to provide rough approximations of these curves, using five values of the QA response for the scenario. To build the utility-response curve, the QA levels for the best-case and worst-case situations are determined. The best-case QA level is the level above which the stakeholders foresee no further utility. For example, if the system responds to the user in 0.1 second, that is perceived as instantaneous; improving it further to make it respond in 0.03 seconds has no utility to the user. Similarly, the worst-case QA level is a minimum threshold above which a system *must* perform; otherwise, the system is of no use to the stakeholders. These best-case and worst-case levels are assigned utility values of 100 and 0, respectively.

Then the *current* and *desired* utility levels for the scenario must be determined. The respective utility values (between 0 and 100) for the current and desired cases are elicited from the stakeholders, with the best-case and worst-case values used as reference points (e.g., we are currently half as good as we would like, but if we reach the desired QA level, then we would have 90% of the maximum utility; hence, the current utility level is set to 50 and the desired utility level is set to 90). In this manner the curves are generated for all the scenarios.

A.3   Inter-Scenario Importance

Different scenarios within a given system have different levels of importance to the stakeholders and therefore different utilities. To characterize the relative importance of each scenario, a weight is assigned through a two-step voting exercise. In the first step, the stakeholders vote on the scenarios to order them. This voting is based on the expected response value for each scenario. Then the stakeholders assign a weight of 1 to the highest-rated scenario and a fractional amount to the other scenarios, based on their relative importance.

If, at some future date, additional scenarios need to be added, they can also be assigned a weight depending on their relative importance. The stakeholders, through consensus, make sure that the scenario weights accord with their intuition.

A.4 Architectural Strategies

The job of the architect or architects is to determine the ASs needed to move from the current QA response level to the desired or even best-case level. A portion of the CBAM is devoted to the enumeration of a set of such ASs. For each strategy, the following information can be derived:

- the expected value of the response in the scenario. The utility of the expected value is calculated using interpolation from the four values already elicited from the stakeholders.
- the effect of the AS on other attributes of interest
- a cost estimate for implementing the AS (an ROI can also be calculated)

A.4.1 Side Effects

Each AS will impact not only the QA from the scenario being considered but also other QAs (which is why there are architectural tradeoffs!). It is important to determine the utility of these additional side effect attribute responses that arise when the AS is applied. In the worst case, a new scenario must be created and its utility-response curve determined by eliciting the best, worst, current, and desired response values from the stakeholders and interpolating the expected utility. In practice, however, if the QA was important to the stakeholders, it would occur in one of the other scenarios, and the utility-response curve for that response would have been constructed already. In that case, the only thing left to be determined would be the expected utility associated with that QA for the given AS. Notice that the expected utility of a particular attribute may be negative if the AS is designed to emphasize an attribute in conflict with it.

Once this additional information has been elicited, the benefit of applying an AS can be calculated by summing the benefits of applying it to all relevant QAs.

A.5 Determining Benefit and Normalization

The overall benefit of an AS can be calculated across scenarios from the utility-response curves by summing the benefit associated with each scenario (weighted by the importance of the scenario). For each AS (i), we calculate a benefit, B_i as follows:

$$B_i = \sum_j (b_{i,j} \times W_j) \qquad \text{Eq. 1}$$

where $b_{i,j}$ is the benefit accrued to AS (i) due to its effect on scenario j, and W_j is the weight of scenario j. Referring to Figure 3, each $b_{i,j}$ is calculated as the change in utility that is brought about by the AS in this scenario. This "delta utility" is calculated as follows:

$b_{i,j} = U_{expected} - U_{current}$, the utility of the expected value of the AS minus the utility of the current system relative to this scenario. The effect of multiplying the weight W_j is to normalize this utility value by the relative importance of the various scenarios.

A.6 Cost

To use the CBAM in making investment decisions, the benefit information needs to be coupled with cost estimates. For each AS under consideration, a cost of implementing that strategy is calculated using a cost model that is appropriate for the system and environment being developed. The CBAM is independent of any particular cost model; no cost model is prescribed.

A.7 Calculating ROI

The ROI value for each AS is the ratio of the total benefit, B_i, to the cost, C_i, of implementing the strategy, as shown in Equation 2.

$$R_i = \frac{B_i}{C_i}$$

Eq. 2

Using this ROI score, the ASs can be rank ordered, and this rank ordering can be used to determine a desirable order for implementing the various strategies. Also note that it might be important to take the shape of the curve into consideration when ranking ASs. Consider curves (a) and (b) in Figure 3. Curve (a) "flattens out" as the QA response improves. In this case, it is likely that a point is reached past which ROI decreases as the QA response improves. In other words, spending more money will not yield a significant increase in utility. On the other hand, consider curve (b) for which a small improvement in QA response can yield a very significant increase in utility. In this case the ROI might increase with improvements in the QA response. Therefore an AS whose ROI is too low to consider might rank significantly higher with a modest improvement in its QA response.

A.8 Dealing with Uncertainty

Within the CBAM, there are a number of uncertain variables that are estimates of stakeholder judgment:

1. inter-scenario weights—This variable defines the relative importance of one scenario with respect to all others. The relative importance is quantified either as votes or as absolute ratios (e.g., the lowest priority scenario has a vote of 1 and the other scenarios have values

greater than 1, indicating their relative values). These weights are dimensionless numbers greater than 0.

2. intra-scenario utility—This is depicted by a utility function, for a particular scenario, for varying levels of the stimulus/response characteristic. (Currently five levels are used: worst, current, expected, desired, and best.) The unit is *utils* and defined between 0 and 100.

3. QA level realized through implementation of an AS—This variable defines the estimated response characteristic that can be achieved by implementing an AS. The dimension of this number depends on the characteristic being measured.

4. cost of implementing an AS—This variable defines the cost of implementing an AS, measured in person-months or dollars.

Given the fact that the values elicited for these variables are certain to be uncertain, we will eventually obtain an ROI score, R(i), that is uncertain. The score (for each AS) can thus be represented by a probability distribution —either uniform, triangular, or normal. R(i) will be the mean/mode value, R(i,max) and R(i,min) will be the bounds of the uniform/triangular distribution, and $\sigma(i)$ will be the standard deviation for a normal distribution. Along with the rank ordering, the certainty/uncertainty regarding this rank ordering can be quantified. The probability that R(i) is greater than R(j) can be calculated according to:

$$\text{prob}(AS_i > AS_j) = \int_{-\infty}^{\infty} f_i(x)F_j(x)dx \qquad \text{Eq. 3}$$

where $f_i(x)$ is the probability density function of R(i) and

$$F_j(x) = \int_{-\infty}^{x} f_j(a)da \qquad \text{Eq. 4}$$

References

[Asundi 01]   Asundi, J.; Kazman, R.; & Klein, M. *Using Economic Considerations to Choose Amongst Architecture Design Alternatives* (CMU/SEI-2001-TR-035, ADA399151). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. <http://www.sei.cmu.edu/publications/documents/01.reports /01tr035.html>.

[Clements 01]   Clements, P.; Kazman, R.; & Klein, M. *Evaluating Software Architectures: Methods and Case Studies.* Boston, MA: Addison-Wesley, 2001.

[Kazman 01]   Kazman, R.; Asundi, J.; & Klein, M. "Quantifying the Costs and Benefits of Architectural Decisions," 297-306. *Proceedings of the 23rd International Conference on Software Engineering (ICSE 23).* Toronto, Canada, May 12-19, 2001. Los Alamitos, CA: IEEE Computer Society, 2001.

[Kazman 99]   Kazman, R.; Barbacci, M.; Klein, M.; Carriere, S. J.; & Woods, S. G. "Experience with Performing Architecture Tradeoff Analysis," 54-63. *Proceedings of the 21st International Conference on Software Engineering (ICSE 21).* Los Angeles, California, May 16-22, 1999. New York, NY: Association for Computing Machinery, 1999.

REPORT DOCUMENTATION PAGE

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (leave blank)	2. REPORT DATE September 2002	3. REPORT TYPE AND DATES COVERED Final

4. TITLE AND SUBTITLE Making Architecture Design Decisions: An Economic Approach	5. FUNDING NUMBERS C — F19628-00-C-0003
6. AUTHOR(S) Rick Kazman, Jai Asundi, Mark Klein	

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2002-TR-035
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	10. SPONSORING/MONITORING AGENCY REPORT NUMBER ESC-TR-2002-035

11. SUPPLEMENTARY NOTES

12.a DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS	12.b DISTRIBUTION CODE

13. ABSTRACT (maximum 200 words)

The resources available to build any system are finite. The decisions involved in building any nontrivial system are complex and typically involve many stakeholders, many requirements, and many technical decisions. The stakeholders have an interest in ensuring that good design decisions are made—decisions that meet their technical objectives and their tolerance for risk. These decisions should, as much as possible, maximize the benefit that the system provides and minimize its cost. The Cost Benefit Analysis Method (CBAM) was created to provide some structure to this decision-making process. The CBAM analyzes architectural decisions from the perspectives of cost, benefit, schedule, and risk. While the CBAM does not make decisions for the stakeholders, it does serve as a tool to inform managers and to structure the inquiry so that rational decisions can be made. This report describes the steps of the CBAM and its application to a real-world system.

14. SUBJECT TERMS architecture design, architecture analysis, economic modeling, decision support	15. NUMBER OF PAGES 42
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL